40

July 1976 Volume 4 Number 7

Popular Computing

In the familiar Fibonacci sequence, the units position has been marked off. This sub-sequence repeats on a cycle of 60, as can be readily verified in a few minutes.

Let us generalize this single-digit sequence as follows:

1	1	← starting values
1	2	
3	4	
6	9	
3	9	
8	1	
0	8	
9	9	
7	6	
5	2	
8		

starting with two columns, both initialized to <u>one</u>, and with <u>one</u> as the next term of the first column. Then each successive term is formed by adding the preceding two terms, alternating columns as shown. Only one digit is retained in each addition (i.e, all arithmetic is modulo 10).

This pattern, too, must repeat, and it does, on a cycle of 217 terms (that is, 217 rows of the 2-column pattern).

Continued on page 3

	1
	1
	2
	3
	5
	8
1	3
2	1
3	4
5	5
8	9
1 4	4
2 3	3
3 7	7
6 1	0
	7
	7
	4
	1
	5
	6
	1
	7
	8
	5
	3
	8
	1

N-SERIES 40

Log 40	1.602059991327962390427477789448986053536379762924217
ln 40	3.688879454113936302852455697600717343752101757349283
$\sqrt{40}$	6.324555320336758663997787088865437067439110278650434
$\sqrt[3]{40}$	3.419951893353393978706217745087720219736110221086110
$\sqrt[10]{40}$	1.446125549591924767921929457440768324506868042667413
$\sqrt[100]{40}$	1.037577630125775761680909013838232479655857286831469
e^{40}	235385266837019985.4078999107490348045088716172545554 6723665125118928916352581695433673
π^{40}	76912142205157127257.26518792378931273281851141229290 96755619735381502311305049350126
$\tan^{-1} 40$	1.545801533175976459729604317990079734205031907185705

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $20.50 per year, or $17.50 if remittance accompanies the order. For Canada and Mexico, add $1.50 to the above rates. For all other countries, add $3.50 per year to the above rates. Back issues $2.50 each. Copyright 1976 by POPULAR COMPUTING.

Editor: Audrey Gruenberger
Publisher: Fred Gruenberger
Associate Editors: David Babcock
Irwin Greenwald

Contributing editors: Richard Andree
William C. McGee
Thomas R. Parkin

Advertising Manager. Ken W. Sims
Art Director: John G. Scott
Business Manager: Ben Moore

PROBLEM 133

CONTEST 9

K-COLUMN FIBONACCI

Extending this scheme to three columns, we have:

starting values	1	1	1
	1	2	3
	4	5	7
	0	4	9
	6	6	0
	9	5	1
	1	0	5
	6	7	7
	2	8	5
	2	4	2

and the 3-column pattern is found to repeat on a cycle of 520 terms. The cycle lengths for the 4- and 5-column patterns are readily obtained, and we have the following table:

T

Number of columns, K	Cycle length
1	60
2	217
3	520
4	42
5	196812

indicating that we have an irregular (indeed, weird) function.

Our 9th contest, then, will award our usual $25 prize to the person who extends table T the furthest. The computer program used must be furnished. All entries must be received by POPULAR COMPUTING, Box 272, Calabasas, California 91302, by September 30, 1976.

ART OF COMPUTING 10 — FJG

When Numerical Methods are devised to implement the techniques of Numerical Analysis, troubles arise from several sources:

1. In a computer, most numbers do not exist. In a binary machine, the numbers .1, pi, and the square root of 5 do not exist, for example. If 8-digit scientific notation is used, there are just 10^8 numbers between zero and one that can be expressed exactly; all others are approximations. Multiple precision does not solve this difficulty, although it may relieve it.

2. The numbers that do exist in a machine (again in scientific notation) are not uniformly dense. There are also just 10^8 numbers between 10^8 and 10^9, which means that in that range the numbers are spread out thinner.

3. The notion that A+B = B+A doesn't always hold. Consider the addition (in a 3-digit system) of:

```
100.
  .1 )
  .1 |
  .1 }     a million of these
  .1 |
  .1 |
  .1 )
```

If the addition is done from the top down, the result is 100. If it is done from the bottom up, the result is an overflow.

4. We can lose significance at any time, with no warning. The addition of:

```
 34567891 E06
-34561234 E06
```

(both correct to 8 significant digits) will be

66570000 E02.

The loss of significance is due entirely to the fact that the numbers are close to each other, but the point is that that loss takes place unseen.

5. Due to the finite nature of all numbers in a computer, problems that are mathematically sound may be computationally unstable. The value of the determinant:

$$\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 10 & 20 & 30 \end{vmatrix}$$

is clearly zero, since the first and third rows are proportional. Mathematically, the determinant:

$$\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 10 & 20 & 29.999999 \end{vmatrix}$$

is non-singular, but in a computer it is "almost singular," which may cause serious trouble when the situation is better disguised than it is here.

6. Rounding errors do not always balance out. See, for example, the calculations on the perimeters of circumscribed polygons given in Richard Hamming's article "Archimedes and the Value of Pi" in our issue No. 12. Or, see the evaluations of the Taylor series for sine in *Numerical Methods and Fortran Programming* (McCracken and Dorn, page 89) in which the value for sine of 2190° is given by the obvious Fortran program as 25902480. Further interesting examples are found in the article "Some Dangers of Machine Calculations" by Leon Winslow in the *Journal of Recreational Mathematics*, Vol. 8, No. 2, page 83.

Despite all this, most numerical processes *do* work and therein lies the real difficulty. When one applies a numerical process to a problem with a known (analytic) result, and that result is obtained (involving thousands of individual calculations), the troubles and pitfalls listed above seem to be remote; that is, they seem to apply only to unusual cases which can't happen to me.

The Error Amplification problem (in Issue No. 32) was designed specifically to reveal such troubles--but it didn't. The expression:

$$2^{\frac{3 \quad 5 \quad 7 \quad 9 \quad 11 \quad 13 \quad 15 \quad 17 \quad \cdots\cdots \quad 99}{2 \quad 4 \quad 6 \quad 8 \quad 10 \quad 12 \quad 14 \quad 16 \quad \cdots\cdots \quad 98}}$$

was to be evaluated by many distinct methods (and, in particular, by taking the various powers and roots in sequence), with the thought that each different method would yield a distinct result. The result was somewhat the opposite; all the approaches yielded essentially the same result, within the limits of precision used. For example, the "true" result may be taken as 248.81398019578 (done with 100-digit arithmetic, working first on the long exponent and then calculating the power by logarithms). Carrying out the calculation sequentially on a pocket calculator, using 12-digit arithmetic, the result is 248.8139762. The result has, if anything, reinforced our intuitive belief that all is well with the world. Providence again seems to be at work guiding fools, drunks, and numerical workers.

Let us explore the theories involved here by doing some numerical integration by the method probably most widely used; namely, the formula of Simpson.

Simpson's Rule provides a widely used method for numeric evaluation of integrals. The rule is:

$$I = \int_a^b f(x)dx \sim \frac{h}{3}\left[y_0 + y_n + 4(y_{odd}) + 2(y_{even})\right]$$

where the interval from a to b is divided into an even number of sub-intervals of width h, and the y values in the formula are the values of the function $f(x)$. The theory assures us of an exact value for polynomials of degree three or less.

As with any new tool, we try it out first on known cases. Thus, for the curve

$$y = x^3 - 11x^2 + 4x + 60$$

the area between -1 and +3 should be obtainable precisely by Simpson's Rule using any even number of sub-intervals, and the result should agree with the analytic solution:

$$x^4/4 - (11x^3)/3 + 2x^2 + 60x$$

evaluated at +3 and -1; the result is 173 1/3 square units.

We can apply the Rule with just two intervals ($N = 2$), and calculate

$$I = \frac{3 - (-1)}{2 \cdot 3} \left[44 + 0 + 4(54) \right]$$

where $h = 4/2$; 44 is the value of the function at $x = -1$; 0 is the value of the function at +3; and 54 is the value of the function at the midpoint of the range (where $x = +1$).

The results agree; our new tool tests out exactly for a known case. We can proceed to try it out in other situations to see how powerful it is. For example, we can construct a 4th degree curve with roots at -1, 1, 9, and 10:

$$y = (x+1)(x-1)(x-9)(x-10)$$

$$y = x^4 - 19x^3 + 89x^2 + 19x - 90$$

and find the area of the large arch (between 1 and 9) analytically to be 2286.9333. Then we can apply Simpson's Rule to the integral, using 2 intervals, then 4 intervals, 6 intervals, 8 intervals, and so on, to find at what point the process gives a reasonable approximation to the desired area. The results are as follows:

Number of intervals	Value by Simpson
2	2560.00000000
4	2304.00000000
6	2290.30452681
8	2288.00000000
10	2287.37024021
20	2286.96063983
30	2286.93872714
50	2286.93403232
70	2286.93351483
100	2286.93337786
120	2286.93335497

Again, we have found that our new tool works as it should. We will try it once more on a non-polynomial. We will calculate the area of a quarter circle of radius <u>one</u>, which is given by:

$$\int_0^1 \sqrt{1 - x^2}\ dx$$

whose value is $\pi/4$ = .785398163397...

Number of intervals	Value by Simpson
2	.74401694
4	.77089879
8	.78029729
16	.78359942
32	.78476305
64	.78517377
128	.78531885
256	.78537013
512	.78538825
1024	.78539466
2048	.78539692
4096	.78539773
8192	.78539801

Having done all this, we should be convinced that our new tool provides a workable method for numerical integration. For an unknown integral, the only sticky problem is the number of sub-intervals to use to insure the level of precision we seek. Let us now apply the tool to one more integral:

$$\int_{e^{-8}}^{1} \frac{dx}{x}$$

whose value is readily found to be exactly 8. The following are some preliminary results (with all arithmetic carried to 12 significant digits):

Number of intervals	Value
4	250.52203908
6	168.14969914
8	127.04780889
12	86.06241235
16	65.65338565
24	45.35960012
32	35.29512874
48	25.34342609
64	20.44756141
96	15.65971496
128	13.34117182
192	11.12187172
256	10.07933864

and now our faith in the new tool may be shaken. What is going on? What should be done about it? If the next integration we try is not a polynomial, is it one for which the process works, or is it one like this one?

These questions invoke predictable answers from students:

1. Take more intervals.
2. Go to multiple precision--and take more intervals.

If we extend the previous table of results, we find:

Number of intervals	Value
512	8.69555371
700	8.39606266
900	8.24235048
1100	8.15922800
1500	8.07883197
2000	8.03854200
3000	8.01258448
5000	8.00255345
10000	8.00022129

We will not do significantly better with multiple precision, unless we go to ridiculous extremes (and we have already consumed significant amounts of computer time).

The trouble is due to the nature of the (cleverly contrived) function and limits--the function is asymptotic to the y-axis. We might do much better if we were to break up the integral into two parts:

from e^{-8} to .1 and from .1 to 1

Calling these two integrals A and B, we have the following results:

Number of intervals	A value	B value	sum value
4	26.93022039	2.40790097	29.33812136
6	19.07052729	2.34178762	21.41231491
10	12.95445588	2.31197801	15.26643389
16	9.67450751	2.30469044	11.97919795
24	7.97010339	2.30309797	10.27320135
32	7.18125527	2.30276351	9.48401877
64	6.16264271	2.30259756	8.46524027
128	5.81053812	2.30258590	8.11312402
256	5.71705426	2.30258514	8.01963940
512	5.69975126	2.30258510	8.00233635
1024	5.69761527	2.30258509	8.00020036
2048	5.69742899	2.30258509	8.00001408
4096	5.69741581	2.30258509	8.00000090
8192	5.69741494	2.30258510	8.00000005

The main point to all of this is: no numerical procedure should be applied blindly. You must know what the problem is, and the weaknesses of the proposed procedure. Never try to substitute brute force for brains. With the possible exception of a square root subroutine, a pathological case can be found for every numerical procedure for which it will not work. And Elmer's Law says that if you do use some procedure blindly, then the next case you try will certainly be that pathological case.

Problem Solution

Problem 65 (issue 19) was as follows:

The 24 odd primes less than 100 are to be placed on the 24 faces of four cubes, in such a way that

(1) Any toss of the four dice produces a sum that is divisible by 4; or

(2) Any toss of the four dice produces a sum that is not divisible by 4.

Are either of these arrangements possible? If the 24 odd primes are taken to be those from 5 through 101, is either arrangement possible?

If the primes from 5 through 101 are placed on the four dice in this way:

5	53	7	47
13	61	11	59
17	73	19	67
29	89	23	71
37	97	31	79
41	101	43	83
all of the form 4K+1		all of the form 4K+3	

In any order on each die.

then any toss of the four dice will show two of one form and two of the other:

$$\begin{array}{l} 4K_1 + 1 \\ 4K_2 + 1 \\ 4K_3 + 3 \\ 4K_4 + 3 \\ \hline 4K \;\; + 8 \end{array}$$

and the sum is always divisible by 4.

If the primes from 3 through 97 are used, there will be 11 of the form 4K+1 and 13 of the form 4K+3. Then no arrangement is possible for either case. □

CONTEST 4 RESULTS

26	27	28	29			30
25		12	13		14	31
24			4	5	15	
	11	3	1		16	32
23	10		2	6	17	33
22	9	8		7		34
21	20			19	18	35

Contest number 4, Square Spiral, appeared on the cover of issue 35.

Consecutive integers were to be entered into the pattern, beginning as shown here, with a square skipped after every prime, and two squares skipped when two primes fell side by side. The pattern was to be extended and a list of the numbers extending northeast from the center was sought.

The longest such list was produced by Jeffrey Shallit, Princeton, New Jersey, and is reproduced on the following page; it has 370 entries, indicating that along the way Mr. Shallit kept track of over half a million numbers, which included some 45,000 primes. The zero entries indicate a square along the northeast diagonal that remains empty.

The computing problem involved in this contest is not, of itself, very practical. It would be straightforward to attack it using large amounts of storage and CPU time. But it could be done, to the limits that Mr. Shallit pushed it, with not over 1600 words of storage (each at least 20 bits long) and, with careful coding, a machine run of perhaps 10 minutes on a modern machine.

In any event, the problem is now useful as a coding exercise with known results. □

1	7839	31786	72070	128748	201837	291381	397384
5	0	32474	73099	130134	203578	293473	399816
14	0	33166	74150	131528	205326	295572	402275
30	8880	0	75199	132928	207070	297674	404732
54	0	34572	76251	134337	208813	299792	0
84	0	35288	77319	135750	210562	301901	409666
123	9991	36013	78387	137168	212336	304022	412128
168	10369	36743	79468	138592	214116	306144	414621
220	10764	37482	80559	140033	0	308286	417111
277	11165	38232	81644	141473	217703	310435	419599
343	11574	38983	82742	142922	219505	0	422106
416	0	39739	83853	144390	221305	314736	424606
495	0	40511	84974	145857	223128	316901	427131
583	12830	41285	86103	147330	224954	319069	429661
675	13266	42065	87236	148813	226786	321241	432185
777	13712	42847	88376	150291	228622	323427	434718
0	14163	0	89526	0	230454	325622	437264
1000	14626	44456	90677	0	232312	327821	439829
1121	15086	45272	91839	154796	0	330037	442394
1252	15553	46098	93003	156316	236051	332250	444945
1391	16031	46924	94180	157842	237932	0	447537
1532	16517	47763	95354	159387	239813	336719	450131
1683	17010	48605	96544	160918	241699	338946	452718
1844	17509	49449	97746	162473	243596	341202	455336
2010	18016	50302	0	164023	0	343453	457944
2183	18526	51172	100169	165591	247411	345720	460572
2364	19062	52036	101390	167163	249327	347984	463208
2551	19593	0	102607	168753	251257	350258	465822
2743	20131	53806	103853	170332	253194	352533	468463
2949	0	54701	105093	0	255135	354828	471096
3163	21235	55612	106345	173518	257095	357137	473751
3378	21795	56518	107604	175125	259047	359435	476399
3603	22363	57422	108866	176735	261023	361771	479065
3835	0	58345	110132	178366	263005	364088	481742
4073	23525	0	111420	179979	264984	366410	484430
4320	24107	60214	112709	181631	266968	368748	487135
4573	24711	61165	114006	183270	268948	371110	489824
4832	25321	62121	115314	184919	270948	373462	492531
5099	25932	63085	116623	186575	272958	375814	495238
5379	26551	64049	0	188244	274976	378189	497957
5658	27175	65036	119264	189911	276988	380560	500687
5945	0	66015	120601	191598	279024	382950	
6245	28457	67006	121939	0	281068	385336	
0	29105	67994	123286	194981	283110	387739	
6861	29762	69009	124639	0	285159	390141	
7181	30430	70030	125997	198388	287233	0	
7508	31107	71044	127372	200110	289309	394952	

SR-52 Notes

The Owner's Manual for the SR-52 programmable calculator says (page 75):

> The halt command entered from the keyboard when the SR-52 is in the run mode will stop execution of a program and return control to the keyboard. The program counter is left wherever it happened to be at the time of program interruption. Program execution will be resumed at that location when RUN is pressed.

This is literally true, but highly misleading. It could be taken to mean that while the machine is executing a stored program it can be interrupted and then restarted at the point of interruption. This is not so. The HALT key does not act at the end of a <u>command</u>, but at the end of a <u>program step</u>. Thus, such simple commands as

STORE
00
→
07

(store the display at register 07) if HALTed at the point indicated by the arrow will restart, if at all, with disastrous results.

Why would one wish to use HALT while a program is executing? If the execution of a program takes a long time, it might be interesting to monitor it. Or, when debugging and testing a complicated program, one might wish to run it for a ways and then cut in to see how it is doing with the various variables. Or, a program known to work properly and give results every two minutes now has run 20 minutes without results--an interrupt to examine storage contents could reveal troubles (which might not exist--the data has changed) and a RESTART could salvage the 20 minutes of calculation.

There can be many legitimate reasons for using a HALT on a running program. To be sure, if the machine has the printer attachment, most of these situations can be covered by suitable PRINT commands inserted within the program, but the printer sells for almost as much as the calculator itself (currently $250 for the printer vs. $300 for the SR-52). The manual speaks in glowing terms of the virtues of having a printer.

There is a technique for inserting a sense-switch HALT in a program, provided that the program uses no trig functions. The SR-52 is set to radian or degree mode by means of an external slide switch. If the switch is set to radian mode, then the calculation of

(SINE 30 - .5)

will be non-zero. This can be tested in the program, and a HALT can be conditionally programmed which will take effect during execution only when the switch is set to degrees.

We seem to have neglected the consecutive numbering of Problems as they have appeared. The following list will bring the system up to date:

Problem number	Name	Reference
126	Life or Death	PC37-14
127	Peripatetic Jumping Bean	PC38-2
128	K-level Sieve	PC38-7
129	Circuitous Race	PC38-18
130	Ring-a-ding	PC39-1
131	Outguess	PC39-4
132	8 Dice	PC39-17

Summing 7-Card Decks

A deck of cards bears six 3-digit numbers on each card, in columns 1 through 18. It is formed of a number of seven-card sub-decks; each sub-deck is identified by the 3-digit number in columns 1-3 of the first of the seven cards.

As the cards are read, a total is to be formed of the other 41 numbers in each sub-deck. After each sub-deck has been handled, its identifying number and the sum for that deck is to be printed.

Assume that a subroutine is available that will read a card and place its six numbers in words addressed at G, G+1, G+2, G+3, G+4, and G+5. The identifying numbers for the sub-decks are all over 500; all data numbers are less than 500. The end of the full deck is signalled by a sentinel card bearing the number 999 in its first three columns.

The following two sets of seven cards would produce the printed lines shown at the bottom:

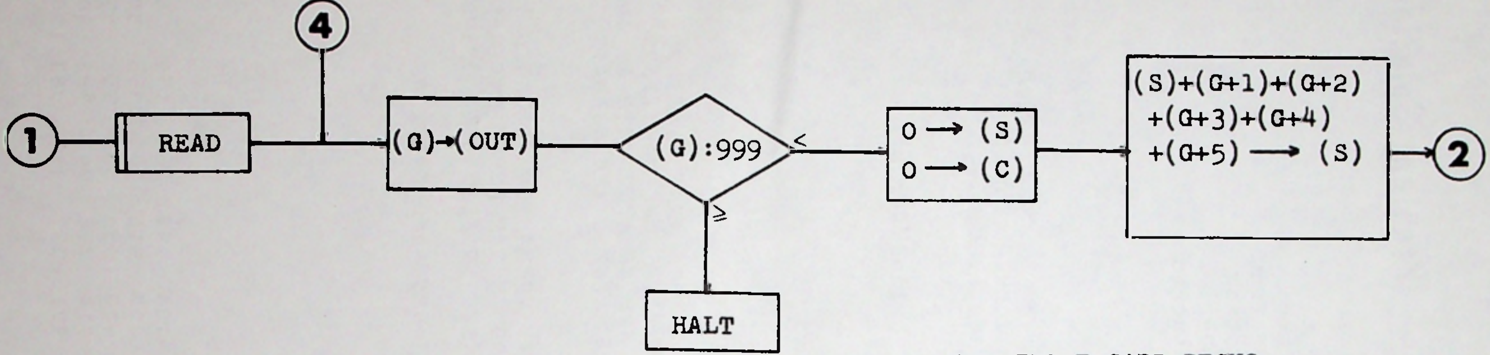
557	001	002	003	004	005
006	007	008	009	010	011
012	013	014	015	016	017
018	019	020	021	022	023
024	025	026	027	028	029
030	031	032	033	034	035
036	037	038	039	040	041

863	100	101	101	102	102
103	103	104	104	105	105
106	106	107	107	108	108
109	109	110	110	111	111
112	112	113	113	114	114
115	115	116	116	117	117
118	118	119	119	120	120

557	861
863	4520

A flowchart for a possible solution to the problem as stated is shown.

This problem is of more than passing interest--it will be referred to in later issues. Notice, in the logic of the proposed solution, that provision has been made for the case in which a sub-deck of less than 7 cards appears.

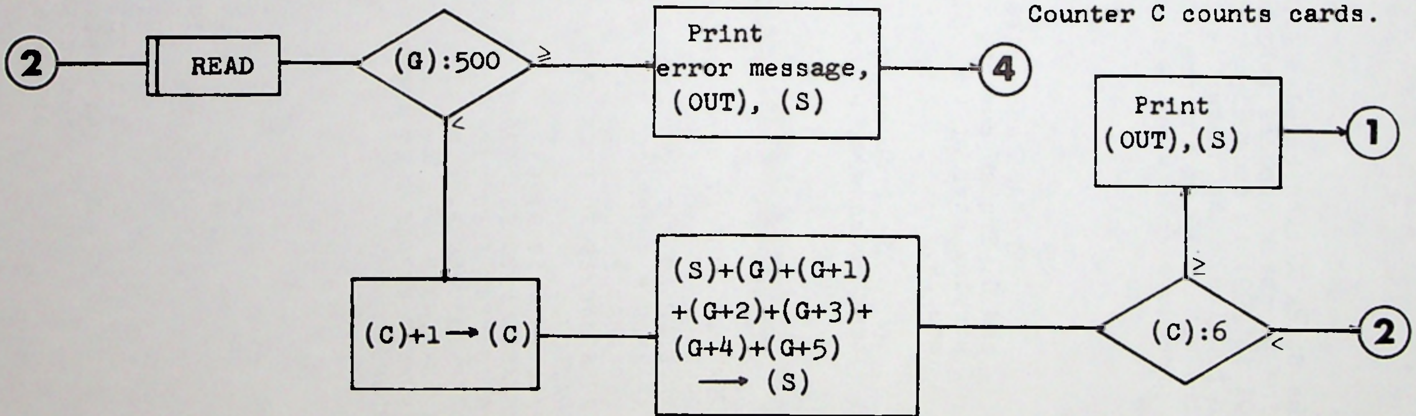


SUMMING 7-CARD DECKS

The subroutine READ inputs the contents of one card to the words G, G+1, G+2, G+3, G+4, G+5.

Required sum is formed in S.

Counter C counts cards.



Problem Solution

The Stack Action Problem (PC37-3) called for a subroutine to stack numbers (which are limited to the range 001 to 999) in ascending order as they arrive from the main routine. No number is to be put in the stack if it lies within 10 of any previously stacked number, and the procedure ends when 50 numbers have been stacked.

The APL code given here comes from M. E. Sandfelder and G. Truman Hunter, Poughkeepsie, New York. The working part of the code is found on lines 2, 8, 12, and 14.

The problem statement in issue 37 called for a test procedure, which is still needed.

```
    ∇ R←BUILD X
[1]   ⍝INITIALIZES RESULT R TO THE VALUE 1000
[2]    R←1000
[3]   ⍝A ONE LINE LOOP THAT PICKS A RANDOM NUMBER UP TO 999 ASSIGNS IT TO
[4]   ⍝A VARIABLE N SUBTRACTS IT FROM ALL VALUES OF R TAKES THE MAGNITUDE
[5]   ⍝OF THE RESULT TESTS TO SEE IF ALL DIFFERENCES ARE GREATER THEN 10 AND
[6]   ⍝IF SO BRANCHES TO NEXT LINE, OTHERWISE STAYS ON LINE 8 AND REPEATS
[7]   ⍝OPERATION.
[8]    →(∨/10≥|R-N←?999)/8
[9]   ⍝CATENATES N TO THE RESULT R TESTS TO SEE IF THE NUMBER OF ELEMENTS
[10]  ⍝IN R IS LESS THEN THE REQUIRED NUMBER CONTAINED IN THE DUMMY VARIABLE X,
[11]  ⍝AND IF SO GOES BACK TO LINE 8 ,OTHERWISE BRANCHES TO NEXT LINE OF PROGRAM
[12]   →(X>⍴R←R,N)/8
[13]  ⍝REARRANGE RESULT R IN ASCENDING ORDER USING GRADEUP AND INDEXING
[14]   R←R[⍋R]
    ∇
[15]  ∇
```

```
      RR←BUILDN 50
      5 10⍴RR

  3  34  71  91 104 131 142 154 179  200
213 233 252 270 287 307 318 351 365  386
412 427 439 455 468 498 513 537 555  572
591 625 652 681 702 714 725 748 783  802
817 846 860 876 888 901 937 955 983 1000
```

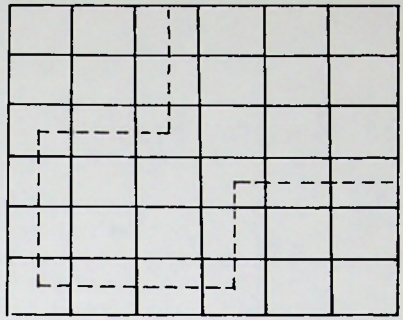
The Gemeroy Problem

In the pattern shown, there are 256 cells, each containing a 3-digit number at random. Some 20 years ago, such a pattern was the basis for newspaper contests, under rules such as these:

> Proceed from the upper left cell (marked A) to the lower right cell (marked B) in straight lines, with each leg at least 3 cells long and not over 6 cells long, with the path not crossing and not touching itself anywhere. The object is to have the sum of the contents of the cells traversed the greatest possible.

A	172	366	939	572	840	237	761	838	158	592	568	132	541	947	595	602	
	568	222	762	742	175	194	629	131	761	063	698	669	765	996	803	898	
	452	935	724	822	198	903	582	202	722	218	430	868	241	269	919	906	
	371	922	027	018	364	556	790	809	974	848	044	962	137	458	394	261	
	872	619	966	762	581	406	034	074	397	764	593	116	885	158	050	423	
	799	769	110	181	556	646	809	929	306	129	115	370	016	837	020	775	
	966	063	808	536	009	684	114	145	450	549	906	688	595	719	782	072	
	227	760	798	401	659	431	922	245	992	934	570	930	401	110	441	150	
	179	932	603	342	627	904	881	566	744	901	171	632	462	947	039	560	
	009	498,	519	602	523	221	090	561	492	931	287	437	045	305	537	920	
	792	439	920	755	093	969	864	355	259	273	349	810	463	156	976	867	
	759	270	217	994	091	284	097	118	020	867	538	783	203	573	741	279	
	140	132	752	475	726	709	766	294	419	952	537	231	395	119	561	024	
	654	795	092	469	586	645	422	344	571	359	158	136	913	784	336	903	
	802	136	640	897	566	049	118	536	602	561	923	286	632	696	982	409	
	559	024	405	569	214	395	937	785	221	970	995	123	553	749	685	365	B

A possible path is indicated in the figure, with a total of 69100. It should be possible to find paths with larger totals. Notice that the rule about the path not touching itself rules out paths like this:



Finding a path with a larger total may be easy. The difficult problem is a systematic attack to find the path with the largest total. The rules for getting from A to B are sufficiently stringent that the number of possible paths is not very large. Given the 256 cell values in storage, then each path that can be developed geometrically can be applied four different ways to the array, and the required sums can be obtained. The trick is to arrange for the controlling program to make the minor alterations on the paths. For example, in the path shown on the array, from the cell containing the value 867 (row 11, column 16) there are at least 6 other paths possible to point B.

The newspaper contests have disappeared, probably because people started using computers on such problems.

If one programmer can do a task in one day,

two programmers can do it in two days